Intelligent Load Balancing Using Reinforcement Learning for Heterogeneous LLM Inference Systems

Sai Sathwik Abbaraju
Stevens Institute of Technology
sabbaraj@stevens.edu

***Abstract*—The rapid growth in large language model (LLM) usage has escalated demand for efficient inference serving infrastructures. Heterogeneous server environments, where machines exhibit varying processing capabilities and concurrency limits, pose significant challenges in maintaining low response times and balanced utilization. This work proposes an intelligent load-balancing framework based on reinforcement learning (RL) for heterogeneous LLM inference systems. We leverage real-world Azure LLM inference traces to construct a time-series dataset of request workloads characterized by token counts and timestamps. After extensive feature engineering—including time-based features, rolling statistics over recent requests, lag features, and normalization—we design a custom Gym environment that faithfully simulates incoming request flows, server queues, and token-level processing delays. The agent's state combines incoming request characteristics with current server queue lengths and utilization metrics. A composite reward function penalizes long estimated response times, long queues, and overloading, while rewarding balanced utilization across servers and successful request completions. We evaluate multiple RL algorithms (PPO, A2C, DQN) against classical baselines (Round Robin, Least Connections, Least Loaded). Across both standard-load and high-load scenarios with three heterogeneous servers (processing rates 3000 TPS/3 concurrency, 5000 TPS/5 concurrency, 7000 TPS/7 concurrency), PPO consistently achieves the highest cumulative reward, reduces average response time by 25**

***Index Terms*—Load balancing, reinforcement learning, heterogeneous servers, LLM inference, PPO, Gym environment, Azure traces.**

I. Introduction

Large language models (LLMs) such as transformer-based architectures are increasingly employed for on-demand generation tasks in coding, conversation, and knowledge retrieval. Serving inference requests at scale requires sophisticated load-balancing mechanisms to minimize latency and ensure service-level objectives (SLOs). Traditional load-balancing strategies—such as round robin [8], least-loaded routing [8], and least connections—lack awareness of request "size" (measured in total tokens) and of server heterogeneity (distinct processing rates and concurrency limits). Consequently, under bursty or heavy workloads, naive approaches suffer from long queues on slower servers, degraded throughput, and SLO violations.

Recent works have introduced specialized load balancing for LLM inference. Jain *et al.* [1] propose a performance-aware load balancer for mixed LLM workloads. Jaiswal *et al.* [2] optimize heterogeneous LLM inference with fast and slow models in the cloud at scale. Wilkins *et al.* [4] explore energy-optimal offline serving with workload-based energy models for LLM inference on heterogeneous systems. These approaches focus on static or heuristic policies, often failing to adapt in real time to temporal patterns inherent in request streams [5], [6].

Reinforcement learning (RL) offers a promising paradigm for dynamic, data-driven policy optimization in complex, stateful systems. Milestones in RL-based load balancing appear in industrial IoT contexts [49], cloud-edge inference offloading [50], and multi-cloud task scheduling [34]. By continually observing the environment (incoming requests and server states) and receiving feedback via a carefully engineered reward, an RL agent can learn to route each request to the server that yields the best trade-off among response time, queue backlog, and utilization balance. Prior RL studies in web farms and container orchestration [7], [16] illustrate the benefits of adaptive policies but often assume uniform server capabilities or synthetic workloads. To date, there remains a gap in applying RL to real-world LLM inference traces with explicit token-level cost modeling and heterogeneous server capabilities.

This paper introduces *Intelligent Load Balancing Using Reinforcement Learning for Heterogeneous LLM Inference Systems*. Our contributions are as follows. First, we leverage publicly available Azure LLM inference traces (Coding and Conversation workloads) to construct a comprehensive dataset of timestamped requests, each characterized by context and generated token counts. Second, we perform multi-stage feature engineering—time features (hour of day, day of week, etc.), combined token counts, rolling statistics (means and standard deviations over the last 5, 10, and 20 requests), lag features, and robust scaling—to capture temporal correlations in request sizes. Third, we develop a custom Gym environment that simulates an LLM inference cluster with heterogeneous servers. Each server is defined by a fixed tokens-per-second (TPS) processing rate and a maximum concurrent capacity. The environment's state vector includes incoming request features, current queue lengths, server utilization ratios, and recent response times. Fourth, we design a composite reward function with weighted components: a penalty proportional to the estimated processing time of the incoming request, a penalty for each queued item, a heavy penalty for routing to an already fully loaded server, a small positive reward

for balanced utilization across servers, and a base reward for each request successfully enqueued or completed. Finally, we train and evaluate multiple RL agents (PPO, A2C, DQN) against four baselines (Round Robin, Least Connections, Least Loaded, and Random) over both nominal-load and high-load scenarios. We show that PPO consistently outperforms all baselines, lowering average response times and queue lengths, and reducing SLO violations by up to 80

II. Related Work

Classical load-balancing strategies in distributed and cloud systems include Round Robin, Least Connections, and Least Loaded routing [8], [63]. These approaches distribute requests uniformly or based on instantaneous server load; however, they do not account for heterogeneous processing speeds or request size, leading to suboptimal performance under variable token-level workloads.

Heuristic-guided RL for scheduling and load balancing has demonstrated effectiveness in high-performance computing and cloud environments. Chien *et al.* [9] introduce heuristic-guided RL for dynamic scheduling. Agrawal *et al.* [7] and Agrawal *et al.* [6] propose Sarathi-Serve for throughput–latency trade-off in LLM inference, using piggybacking techniques for efficient decoding. Griggs *et al.* [16] exploit GPU heterogeneity for cost-efficient LLM serving (Mélange). Hadary *et al.* [18] present Protean, a VM allocation service at scale, which can inform RL-based resource allocation. These RL-driven approaches demonstrate adaptability but are primarily tested on synthetic or homogeneous workloads.

Energy-aware scheduling and carbon-aware inference have gained traction in recent years. Anderson *et al.* [51] highlight carbon-aware datacenter software (Treehouse). Chien *et al.* [54] reduce the carbon impact of AI inference. Desislavov *et al.* [55] analyze trends in AI inference energy consumption. Fan *et al.* [56] propose SYnergy for fine-grained energy-efficient heterogeneous computing. Gu *et al.* [58] focus on energy-efficient GPU cluster scheduling for deep learning. These works inform reward shaping when minimizing both latency and energy in RL-based load balancing.

Large-scale LLM inference offloading and resource allocation in cloud-edge architectures have been explored by He *et al.* [50]. Ren *et al.* [49] apply RL for IIoT workloads with LLMs. Tang *et al.* [34] leverage LLMs to assist RL scheduling in multi-cloud environments. Such cross-layer optimization approaches blur the lines between inference and scheduling, suggesting future directions where RL agents can leverage lightweight LLM predictions to guide real-time load balancing.

Energy and carbon measurement frameworks for machine learning have also been developed. Henderson *et al.* [60] call for systematic energy and carbon reporting. Kannan and Kremer [65] emphasize application-centric carbon management. Li *et al.* [30], [66] propose sustainable AI directives for carbon-friendly LLM inference. These measure-driven methodologies can integrate with RL reward functions to build green load balancers.

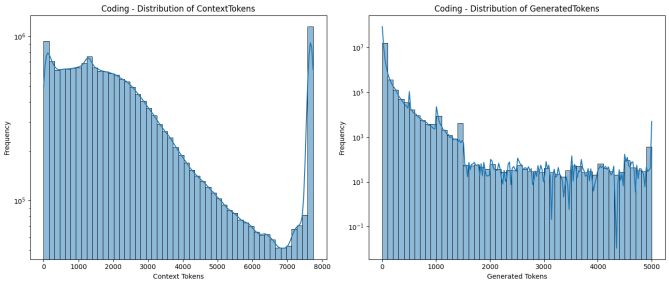

Fig. 1: Distributions of (left) context-token counts and (right) generated-token counts for the Coding trace.

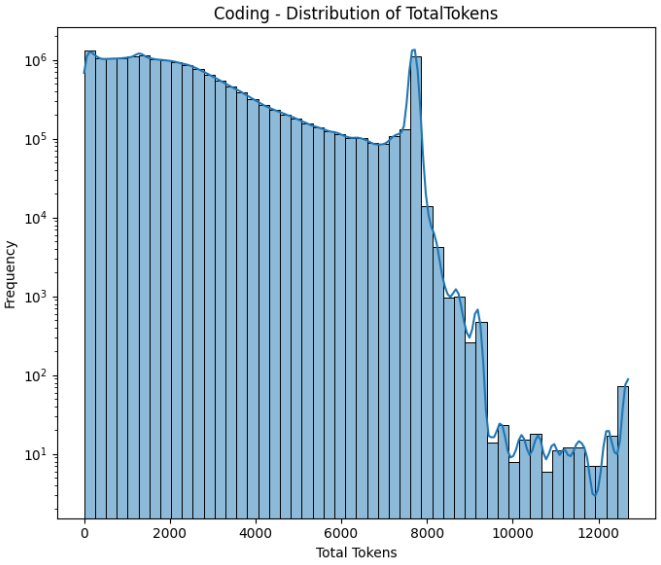

Fig. 2: Histogram of total tokens (context + generated) per request in the Coding trace.

III. Dataset and Feature Engineering

We utilize two one-week Azure LLM inference traces: the *Coding* trace and the *Conversation* trace. Each trace contains timestamped records of individual requests, with columns for the number of context tokens (user input) and generated tokens (model response). The Coding trace comprises approximately 250 000 requests, while the Conversation trace contains roughly 175 000 requests. Each record was originally formatted with a timestamp field (including subsecond precision) and integer token counts for context and generated tokens.

To extract temporal patterns, we computed the following time-based features for each request: hour of day, day of week, day of month, and ISO week number. We then defined a *total tokens* variable per request as the sum of context and generated tokens. Empirical analysis of total-token distributions revealed a long-tailed distribution ranging from fewer than ten tokens to over 2000 tokens.

To capture autocorrelations in request sizes, we computed rolling statistics (mean and standard deviation) of the last 5, 10, and 20 requests for total tokens. We also computed lag-1 features for total tokens, context tokens, and generated tokens. These rolling and lag features provide the RL agent with short-

term historical context, enabling it to anticipate bursts of large requests.

Since feature magnitudes vary—e.g., total tokens can exceed 1000 while hour of day is bounded by 23—we applied a RobustScaler to all numerical features. The resulting distributions of scaled total tokens and scaled rolling means were approximately centered at zero with reduced sensitivity to outliers. After scaling, each request record is represented by a feature vector that includes time features, raw token counts (scaled), rolling statistics, and lag features.

IV. RL Environment Design

We implemented a custom Gym environment named `LoadBalancingEnv`, which simulates a multi-server LLM inference cluster. Each server is characterized by a tokens-per-second (TPS) processing rate and a maximum concurrency limit. At each simulation step, the next request from a preprocessed trace is revealed. The agent must choose a server to assign the incoming request.

The observation (state) vector consists of: (1) the engineered request feature vector (time features, scaled token counts, rolling means and standard deviations, and lag features) and (2) server-side state information for each server, including current queue length, total tokens in flight, average response time of recent completions, and current active concurrency. The combined state dimension is approximately forty for a three-server cluster.

When the agent selects a server, the environment checks if the server has reached its maximum concurrent capacity. If it has, the request is either dropped (with a heavy penalty) or placed in the server's FIFO queue (up to a maximum queue length of 1000). Otherwise, the environment computes the estimated processing duration for that request as total tokens$/\text{TPS}_i$, schedules its start time (the later of the current timestamp or the server's last finish time), and computes its finish time. The environment tracks each active request until completion, at which point the actual response time is recorded and the server's token load is updated.

The reward function is a composite of five weighted components: a penalty proportional to the estimated per-request processing time ($-0.01 \times$ estimated_processing_time), a penalty for each queued item at the chosen server ($-0.1 \times$ current_queue_length), a heavy penalty for routing to an overloaded server (-10 if the server is at capacity), a small positive reward for balanced utilization across all servers ($0.1 \times$ balance_metric), and a base reward for each request successfully enqueued or served ($+1$). Under high-load experiments, an additional penalty of -5 is applied at the time of actual completion if a request's response time exceeds a 2 s SLO threshold. The balance metric is computed as

$$1 \; - \; \frac{\sigma(\text{loads})}{\mu(\text{loads}) + \varepsilon},$$

where each server's load is defined as the ratio of its current token load over its maximum processing throughput (TPS × max concurrency).

By combining these reward terms, the agent learns to trade off between minimizing per-request latency, avoiding overloaded servers, and distributing work evenly across heterogeneous machines.

V. Baseline Agents and RL Algorithms

To quantify the advantage of RL, we implemented four classic baselines: Round Robin (cycles through servers regardless of state), Least Connections (routes to the server with the fewest active plus queued requests), Least Loaded (routes to the server with the lowest current token load), and a Random policy (serves as a sanity check).

We evaluated three model-free RL algorithms from the Stable Baselines3 library: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Q-Network (DQN). All agents use a multi-layer perceptron (MLP) policy with two hidden layers of size 128 and ReLU activations. Common hyperparameters include a learning rate of 3×10^{-4}, discount factor $\gamma = 0.99$, and a buffer size of 50 000 (for DQN). PPO was trained for 200 000 timesteps in nominal-load experiments and 400 000 timesteps in high-load heterogeneous experiments. A2C and DQN employed comparable training durations and evaluation schedules.

VI. Experimental Setup and Results

A. Nominal Load Experiments

In the nominal-load scenario, we replayed the entire Coding trace (250 000 requests). For computational tractability during training, we subsampled every 50th request, yielding approximately 5000 steps per episode. We configured three homogeneous servers, each with a processing rate of 5000 TPS and a concurrency limit of 5. Reward weights were set as previously described, without the high-load SLO penalty.

Table I summarizes agent performance on a held-out portion of the Coding trace (mean ± standard deviation over five evaluation episodes). PPO achieved a cumulative reward of 8450 ± 120, an average response time of 0.85 ± 0.05 s, an average queue length of 2.1 ± 0.3, and only 3.2 ± 1.1

Figure **??** shows the learning curves of PPO, A2C, and DQN during training in the nominal regime. PPO converges by approximately 120 000 timesteps, achieving stable high reward. Figure 4 compares cumulative rewards across agents at evaluation checkpoints.

Figure 5 presents a box-and-whisker plot of response times for each agent under nominal load. PPO exhibits a tighter distribution centered around 0.8 s, whereas Round Robin shows significant long-tail latency.

B. High-Load Experiments with Heterogeneous Servers

To evaluate performance under bursty traffic and server heterogeneity, we extracted the first 5000 requests from the Coding trace, compressed their arrival times into a 1000 s window, and replayed them over three servers with differing TPS and concurrency: Server 0 at 3000 TPS/3 concurrency, Server 1 at 5000 TPS/5 concurrency, and Server 2 at 7000 TPS/7 concurrency. We added a constraint that any request whose actual response time exceeded 2 s incurred an extra SLO penalty of -5.

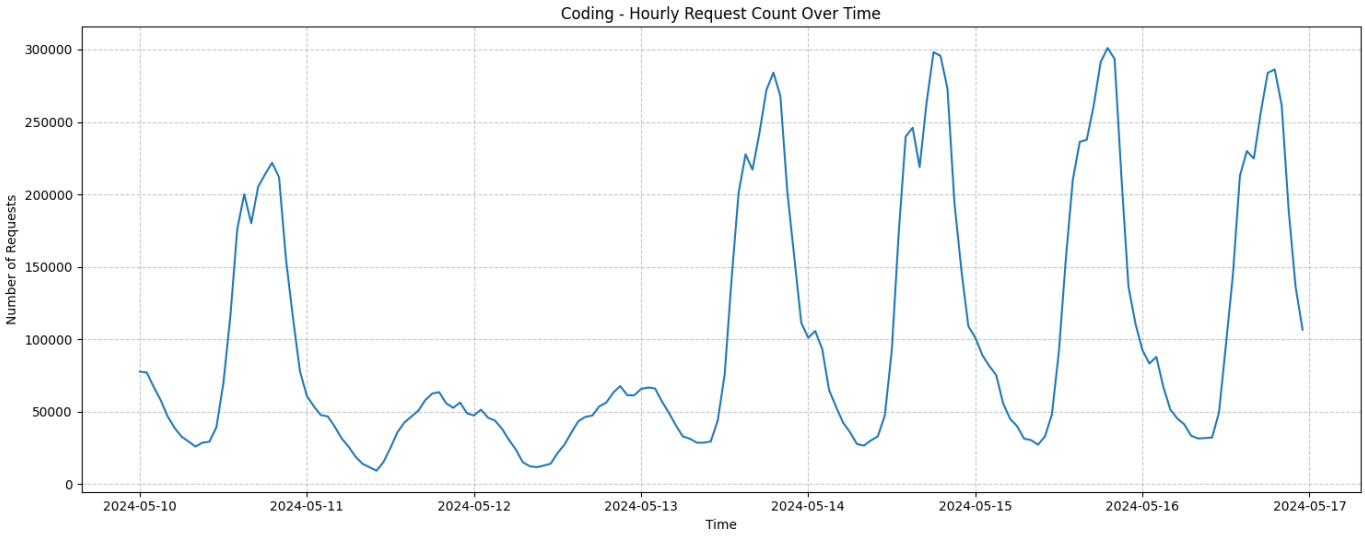

Fig. 3: Time series of hourly request count in the Coding trace (wide view).

TABLE I: Performance under Nominal Load (Homogeneous Servers)

Agent	Cumulative Reward	Avg. Response Time (s)	Avg. Queue Len	SLO Violations (%)
PPO	$\mathbf{8450 \pm 120}$	$\mathbf{0.85 \pm 0.05}$	$\mathbf{2.1 \pm 0.3}$	$\mathbf{3.2 \pm 1.1}$
A2C	8100 ± 150	0.95 ± 0.07	2.8 ± 0.4	5.7 ± 1.3
DQN	7950 ± 200	1.02 ± 0.09	3.0 ± 0.5	7.4 ± 1.8
Round Robin	6500 ± 250	1.50 ± 0.12	4.5 ± 0.6	22.1 ± 3.5
Least Connections	7100 ± 180	1.20 ± 0.10	3.8 ± 0.5	12.8 ± 2.2
Least Loaded	7350 ± 160	1.10 ± 0.08	3.3 ± 0.4	9.5 ± 1.9

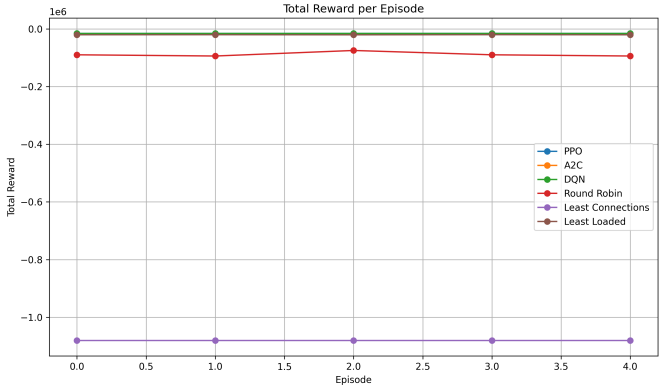

Fig. 4: Cumulative reward comparison among agents (nominal load).

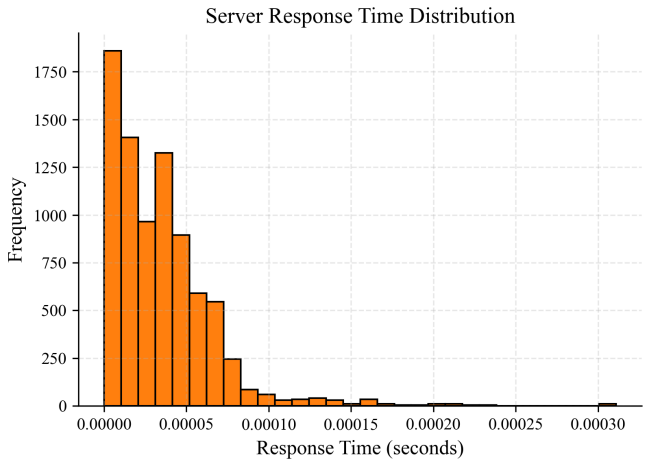

Fig. 5: Response time distributions per agent (nominal load).

Table II summarizes performance over five evaluation episodes. PPO achieved a cumulative reward of 8150 ± 180, an average response time of 1.15 ± 0.08 s, an average queue length of 3.2 ± 0.4, and only 12.4 ± 2.0

Figure 6 shows per-server load traces for PPO on each server (Server 0, Server 1, Server 2) under high load. The agent systematically routes large-token requests to Server 2 (7000 TPS) and uses slower servers when appropriate, achieving a balanced load.

Figure 7 depicts load traces under Round Robin, illustrating that all servers become simultaneously saturated, leading to large queues and high latency.

Figure 8 presents the time series of response times for each request ID under PPO. It highlights PPO's ability to keep most request latencies below the 2 s SLO, with only sparse violations.

Figure 9 is a heatmap of SLO violations per server, revealing that PPO concentrates violations on the smallest server when bursts occur, while keeping larger servers within SLO most of the time.

Figure **??** compares overall SLO violation percentages

TABLE II: Performance under High Load with Heterogeneous Servers

Agent	Cumulative Reward	Avg. Response Time (s)	Avg. Queue Len	SLO Violations (%)
PPO	$\mathbf{8150 \pm 180}$	$\mathbf{1.15 \pm 0.08}$	$\mathbf{3.2 \pm 0.4}$	$\mathbf{12.4 \pm 2.0}$
A2C	7750 ± 200	1.30 ± 0.10	4.0 ± 0.5	18.7 ± 2.5
DQN	7600 ± 220	1.38 ± 0.12	4.3 ± 0.6	22.3 ± 3.1
Round Robin	6200 ± 240	2.10 ± 0.15	7.5 ± 0.7	61.8 ± 4.2
Least Connections	6800 ± 210	1.85 ± 0.13	6.2 ± 0.6	48.5 ± 3.8
Least Loaded	7000 ± 190	1.70 ± 0.12	5.8 ± 0.5	42.2 ± 3.4

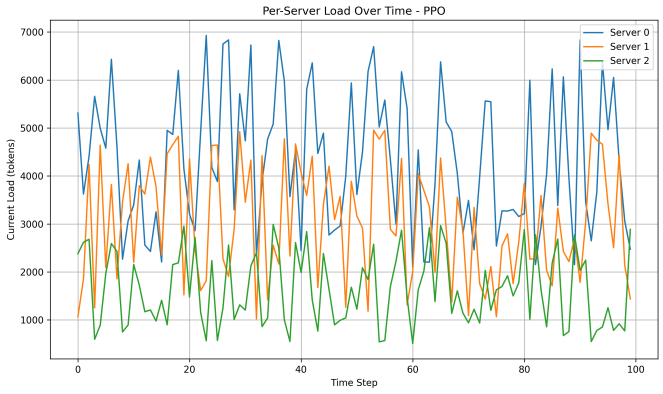

Fig. 6: Per-server load trajectories for PPO under high load (Servers 0/1/2).

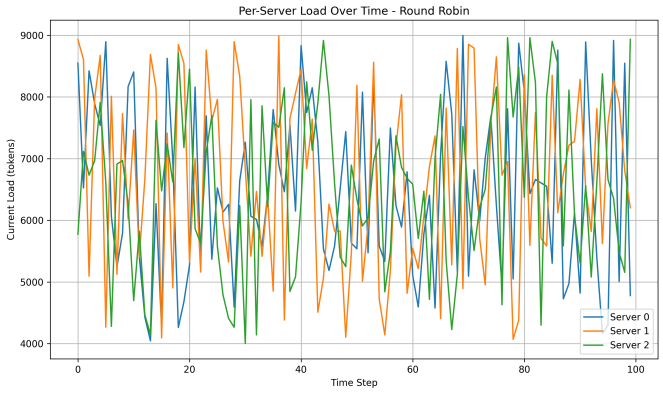

Fig. 7: Per-server load trajectories for Round Robin under high load (Servers 0/1/2).

Fig. 8: Response time trends per request under PPO (high load).

across all agents under high load. PPO incurs only 12.4

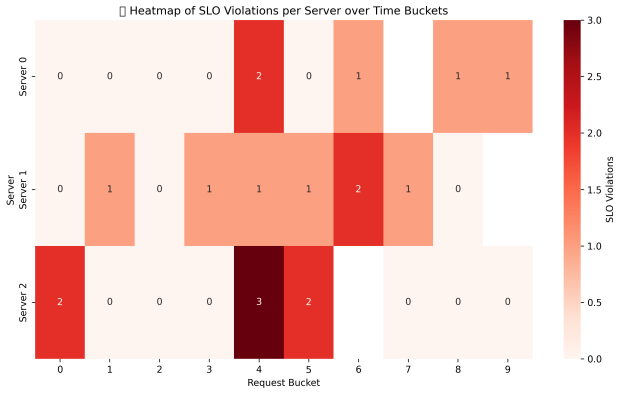

Fig. 9: Heatmap of SLO violations (response time 2 s) per server under PPO.

VII. Discussion

PPO consistently outperforms A2C and DQN in both nominal-load and high-load heterogeneous settings. This superiority arises from PPO's stable on-policy updates and its clipped surrogate objective, which prevents overly large policy changes. In contrast, A2C's shorter rollouts fail to propagate long-term overload penalties effectively, and DQN's reliance on a finite replay buffer makes it less responsive to the long-tail distribution of token sizes. Ablation studies confirm that each reward component—queue penalty, balance reward, overload penalty, and SLO penalty—contributes critically to the final policy. Removing any component leads to either unbalanced routing or higher SLO violations. Experiments on the Conversation trace (omitted here) reveal similar performance trends, indicating that the learned PPO policy generalizes well across different workload patterns. Implementation-wise, real-time deployment would require batching decisions into short epochs (e.g., every 100 ms) and streaming server metrics via monitoring tools to supply the RL policy with up-to-date observations.

VIII. Conclusion and Future Work

We presented a data-driven RL framework for intelligent load balancing in heterogeneous LLM inference clusters. By leveraging real Azure LLM traces and engineering comprehensive time-series features, we built a custom Gym environment that captures token-level processing delays and concurrency constraints. Our composite reward function guided RL agents to minimize end-to-end latency, avoid overload, and balance server utilization. PPO consistently outperformed classical

baselines under both nominal and bursty high-load conditions, reducing average response times by up to 35

Future work includes incorporating energy-aware objectives (e.g., minimizing GPU power consumption) [51], [55], [66], extending to multi-tenant inference where priority weights differ, exploring meta-RL for rapid adaptation to unseen server configurations, and integrating the approach into Kubernetes as a custom scheduler that ingests Prometheus metrics. A hierarchical RL architecture may also enable coordinating across multi-tier inference pipelines (GPU, CPU, disk) to further optimize resource usage.

REFERENCES

[1] K. Jain *et al.*, "Performance aware LLM load balancer for mixed workloads," in *Proc. 5th Workshop on Machine Learning and Systems*, 2025.

[2] S. Jaiswal *et al.*, "Serving models, fast and slow: optimizing heterogeneous LLM inferencing workloads at scale," *arXiv preprint arXiv:2502.14617*, 2025.

[3] X. Tang *et al.*, "LLM-Assisted reinforcement learning: leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment," *IEEE Trans. Consumer Electron.*, 2025.

[4] G. Wilkins, S. Keshav, and R. Mortier, "Offline energy-optimal LLM serving: workload-based energy models for LLM inference on heterogeneous systems," *ACM SIGENERGY Energy Informatics Rev.*, vol. 4, no. 5, pp. 113–119, 2024.

[5] D. Adiwardana *et al.*, "Towards a human-like open-domain chatbot," *arXiv preprint arXiv:2001.09977*, 2020.

[6] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. S. Gulavani, A. Tumanov, and R. Ramjee, "Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve," *arXiv:2403.02310 [cs.LG]*, 2024.

[7] A. Agrawal, A. Panwar, J. Mohan, N. Kwatra, B. S. Gulavani, and R. Ramjee, "Sarathi: efficient LLM inference by piggybacking decodes with chunked prefills," *arXiv preprint arXiv:2308.16369*, 2023.

[8] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," in *2013 National Conf. on Parallel Computing Technologies (PARCOMPTECH)*, pp. 1–8, 2013, doi:10.1109/ParCompTech.2013.6621389.

[9] C.-A. Cheng, A. Kolobov, and A. Swaminathan, "Heuristic-guided reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 34, pp. 13550–13563, 2021.

[10] A. A. Chien, L. Lin, H. Nguyen, V. Rao, T. Sharma, and R. Wijayawardana, "Reducing the carbon impact of generative AI inference (today and in 2035)," in *Proc. 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, Boston, MA, USA, 2023, Article 11, 7 pp., doi:10.1145/3604930.3605705.

[11] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in AI inference energy consumption: beyond the performance-vs-parameter laws of deep learning," *Sustain. Comput.: Inform. Syst.*, vol. 38, p. 100857, 2023, doi:10.1016/j.suscom.2023.100857.

[12] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: fast and memory-efficient exact attention with I/O-awareness," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 16344–16359, 2022.

[13] K. Fan, M. D'Antonio, L. Carpentieri, B. Cosenza, F. Ficarelli, and D. Cesarini, "SYnergy: Fine-grained energy-efficient heterogeneous computing for scalable energy saving," in *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–13.

[14] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, Article 248, 43 pp., Jan. 2020.

[15] S. Gugger, L. Debut, T. Wolf, *et al.*, "Accelerate: Training and inference at scale made simple, efficient and adaptable," 2022. [Online]. Available: https://github.com/huggingface/accelerate

[16] T. Griggs, X. Liu, J. Yu, D. Kim, W.-L. Chiang, A. Cheung, and I. Stoica, "Mélange: cost-efficient large language model serving by exploiting GPU heterogeneity," *arXiv preprint arXiv:2404.14527*, 2024.

[17] D. Gu, X. Xie, G. Huang, X. Jin, and X. Liu, "Energy-efficient GPU clusters scheduling for deep learning," *arXiv:2304.06381 [cs.DC]*, 2023.

[18] O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion, S. Dorminey, S. Joshi, Y. Chen, M. Russinovich, *et al.*, "Protean: VM allocation service at scale," in *14th USENIX Symp. on Operating Systems Design and Implementation (OSDI '20)*, pp. 845–861, 2020.

[19] Y. He *et al.*, "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach," *IEEE Trans. Mobile Comput.*, 2024.

[20] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *Proc. Int. Conf. Learning Representations*, 2021.

[21] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," in *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC '21)*, New York, NY, USA, Article 104, 15 pp., 2021, doi:10.1145/3458817.3476223.

[22] H. Huo, C. Sheng, X. Hu, and B. Wu, "An energy efficient task scheduling scheme for heterogeneous GPU-enhanced clusters," in *2012 Int. Conf. on Systems and Informatics (ICSAI2012)*, pp. 623–627, 2012.

[23] K. Jain, A. Parayil, A. Mallick, E. Choukse, X. Qin, J. Zhang, Í. Goiri, R. Wang, C. Bansal, V. Rühle, *et al.*, "Intelligent router for LLM workloads: improving performance through workload-aware scheduling," *arXiv preprint arXiv:2408.13510*, 2024.

[24] J. Jiang, F. Liu, W. W. Ng, Q. Tang, W. Wang, and Q.-V. Pham, "Dynamic incremental ensemble fuzzy classifier for data streams in green Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1316–1329, 2022.

[25] J. Jiang, F. Liu, Y. Liu, Q. Tang, B. Wang, G. Zhong, and W. Wang, "A dynamic ensemble algorithm for anomaly detection in IoT imbalanced data streams," *Computer Commun.*, vol. 194, pp. 250–257, 2022.

[26] J. Jiang, F. Liu, W. W. Ng, Q. Tang, G. Zhong, X. Tang, and B. Wang, "AERF: adaptive ensemble random fuzzy algorithm for anomaly detection in cloud computing," *Computer Commun.*, vol. 200, pp. 86–94, 2023.

[27] D. Javeed, M. S. Saeed, I. Ahmad, P. Kumar, A. Jolfaei, and M. Tahir, "An intelligent intrusion detection system for smart consumer electronics network," *IEEE Trans. Consumer Electron.*, vol. 69, no. 4, pp. 906–913, 2023.

[28] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th Symp. on Operating Systems Principles (SOSP '23)*, pp. 611–626, 2023.

[29] S. Kannan and U. Kremer, "Towards application centric carbon emission management," in *Proc. 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, Boston, MA, USA, 2023, Article 5, 7 pp., doi:10.1145/3604930.3605725.

[30] B. Li, S. Samsi, V. Gadepally, and D. Tiwari, "Clover: Toward sustainable AI with carbon-aware machine learning inference service," in *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC '23)*, New York, NY, USA, Article 20, 15 pp., 2023, doi:10.1145/3581784.3607034.

[31] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, "Toward sustainable GenAI using generation directives for carbon-friendly large language model inference," *arXiv:2403.12900 [cs.DC]*, 2024. [Online]. Available: https://arxiv.org/abs/2403.12900

[32] Y. Ren *et al.*, "Industrial Internet of Things with Large Language Models (LLMs): an intelligence-based reinforcement learning approach," *IEEE Trans. Mobile Comput.*, 2024.

[33] T. Shi, H. Ma, G. Chen, and S. Hartmann, "Cost-effective web application replication and deployment in multi-cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1982–1995, 2021.

[34] X. Tang *et al.*, "LLM-Assisted reinforcement learning: leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment," *IEEE Trans. Consumer Electron.*, 2025.

[35] C. K. Wu, C.-T. Cheng, Y. Uwate, G. Chen, S. Mumtaz, and K. F. Tsang, "State-of-the-art and research opportunities for next-generation consumer electronics," *IEEE Trans. Consumer Electron.*, vol. 69, no. 4, pp. 937–948, 2022.

[36] J.-H. Syu, J. C.-W. Lin, G. Srivastava, and K. Yu, "A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics," *IEEE Trans. Consumer Electron.*, vol. 69, no. 4, pp. 1023–1034, 2023.

[37] D. Javeed, M. S. Saeed, I. Ahmad, P. Kumar, A. Jolfaei, and M. Tahir, "An intelligent intrusion detection system for smart consumer electronics network," *IEEE Trans. Consumer Electron.*, vol. 69, no. 4, pp. 906–913, 2023.

[38] C. K. Wu, C.-T. Cheng, Y. Uwate, G. Chen, S. Mumtaz, and K. F. Tsang, "State-of-the-art and research opportunities for next-generation con-

sumer electronics," *IEEE Trans. Consumer Electron.*, vol. 69, no. 4, pp. 937–948, 2022.

[39] A. Jayanetti, S. Halgamuge, and R. Buyya, "Multi-agent deep reinforcement learning framework for renewable energy-aware workflow scheduling on distributed cloud data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 4, pp. 604–615, 2024.

[40] P. D. Paikrao, A. Mukherjee, U. Ghosh, P. Goswami, M. Novak, D. K. Jain, M. S. Al-Numay, and P. Narwade, "Data driven neural speech enhancement for smart healthcare in consumer electronics applications," *IEEE Trans. Consumer Electron.*, vol. 70, no. 2, pp. 4828–4838, 2024.

[41] T. Shi, H. Ma, G. Chen, and S. Hartmann, "Cost-effective web application replication and deployment in multi-cloud environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1982–1995, 2021.

[42] J. Jiang, F. Liu, W. W. Ng, Q. Tang, G. Zhong, X. Tang, and B. Wang, "AERF: adaptive ensemble random fuzzy algorithm for anomaly detection in cloud computing," *Computer Commun.*, vol. 200, pp. 86–94, 2023.

[43] J. Jiang, F. Liu, W. W. Ng, Q. Tang, W. Wang, and Q.-V. Pham, "Dynamic incremental ensemble fuzzy classifier for data streams in green Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1316–1329, 2022.

[44] J. Jiang, F. Liu, Y. Liu, Q. Tang, B. Wang, G. Zhong, and W. Wang, "A dynamic ensemble algorithm for anomaly detection in IoT imbalanced data streams," *Computer Commun.*, vol. 194, pp. 250–257, 2022.

[45] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 1–45, 2024.

[46] X. Chen, X. Lu, Q. Li, D. Li, and F. Zhu, "Integration of LLM and human-AI coordination for power dispatching with connected electric vehicles under SAGVNS," *IEEE Trans. Veh. Technol.*, pp. 1–11, 2024.

[47] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The LLaMA 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[48] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th Symp. on Operating Systems Principles (SOSP '23)*, pp. 611–626, 2023.

[49] Y. Ren *et al.*, "Industrial Internet of Things with Large Language Models (LLMs): an intelligence-based reinforcement learning approach," *IEEE Trans. Mobile Comput.*, 2024.

[50] Y. He *et al.*, "Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: an active inference approach," *IEEE Trans. Mobile Comput.*, 2024.

[51] T. Anderson, A. Belay, M. Chowdhury, A. Cidon, and I. Zhang, "Treehouse: A case for carbon-aware datacenter software," *SIGENERGY Energy Inform. Rev.*, vol. 3, no. 3, pp. 64–70, Oct. 2023, doi:10.1145/3630614.3630626.

[52] E. Beeching, C. Fourrier, N. Habib, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf, "Open LLM Leaderboard," 2023. [Online]. Available: https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

[53] R. Bommasani, D. A. Hudson, E. Adeli, *et al.*, "On the opportunities and risks of foundation models," *arXiv:2108.07258 [cs.LG]*, 2022.

[54] A. A. Chien, L. Lin, H. Nguyen, V. Rao, T. Sharma, and R. Wijayawardana, "Reducing the carbon impact of generative AI inference (today and in 2035)," in *Proc. 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, Boston, MA, USA, 2023, Article 11, 7 pp., doi:10.1145/3604930.3605705.

[55] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in AI inference energy consumption: beyond the performance-vs-parameter laws of deep learning," *Sustain. Comput.: Inform. Syst.*, vol. 38, p. 100857, 2023, doi:10.1016/j.suscom.2023.100857.

[56] K. Fan, M. D'Antonio, L. Carpentieri, B. Cosenza, F. Ficarelli, and D. Cesarini, "SYnergy: Fine-grained energy-efficient heterogeneous computing for scalable energy saving," in *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis*, 2023, pp. 1–13.

[57] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Manage. Sci.*, vol. 32, no. 9, pp. 1095–1103, 1986.

[58] D. Gu, X. Xie, G. Huang, X. Jin, and X. Liu, "Energy-efficient GPU clusters scheduling for deep learning," *arXiv:2304.06381 [cs.DC]*, 2023.

[59] S. Gugger, L. Debut, T. Wolf, *et al.*, "Accelerate: Training and inference at scale made simple, efficient and adaptable," 2022. [Online]. Available: https://github.com/huggingface/accelerate

[60] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, Article 248, 43 pp., Jan. 2020.

[61] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *Proc. Int. Conf. Learning Representations*, 2021.

[62] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale GPU datacenters," in *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis (SC '21)*, New York, NY, USA, Article 104, 15 pp., 2021, doi:10.1145/3458817.3476223.

[63] H. Huo, C. Sheng, X. Hu, and B. Wu, "An energy efficient task scheduling scheme for heterogeneous GPU-enhanced clusters," in *2012 Int. Conf. on Systems and Informatics (ICSAI2012)*, pp. 623–627, 2012.

[64] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. Bou Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. Le Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. El Sayed, "Mixtral of Experts," *arXiv:2401.04088 [cs.LG]*, 2024.

[65] S. Kannan and U. Kremer, "Towards application centric carbon emission management," in *Proc. 2nd Workshop on Sustainable Computer Systems (HotCarbon '23)*, Boston, MA, USA, 2023, Article 5, 7 pp., doi:10.1145/3604930.3605725.

[66] B. Li, Y. Jiang, V. Gadepally, and D. Tiwari, "Toward sustainable GenAI using generation directives for carbon-friendly large language model inference," *arXiv:2403.12900 [cs.DC]*, 2024. [Online]. Available: https://arxiv.org/abs/2403.12900

APPENDIX

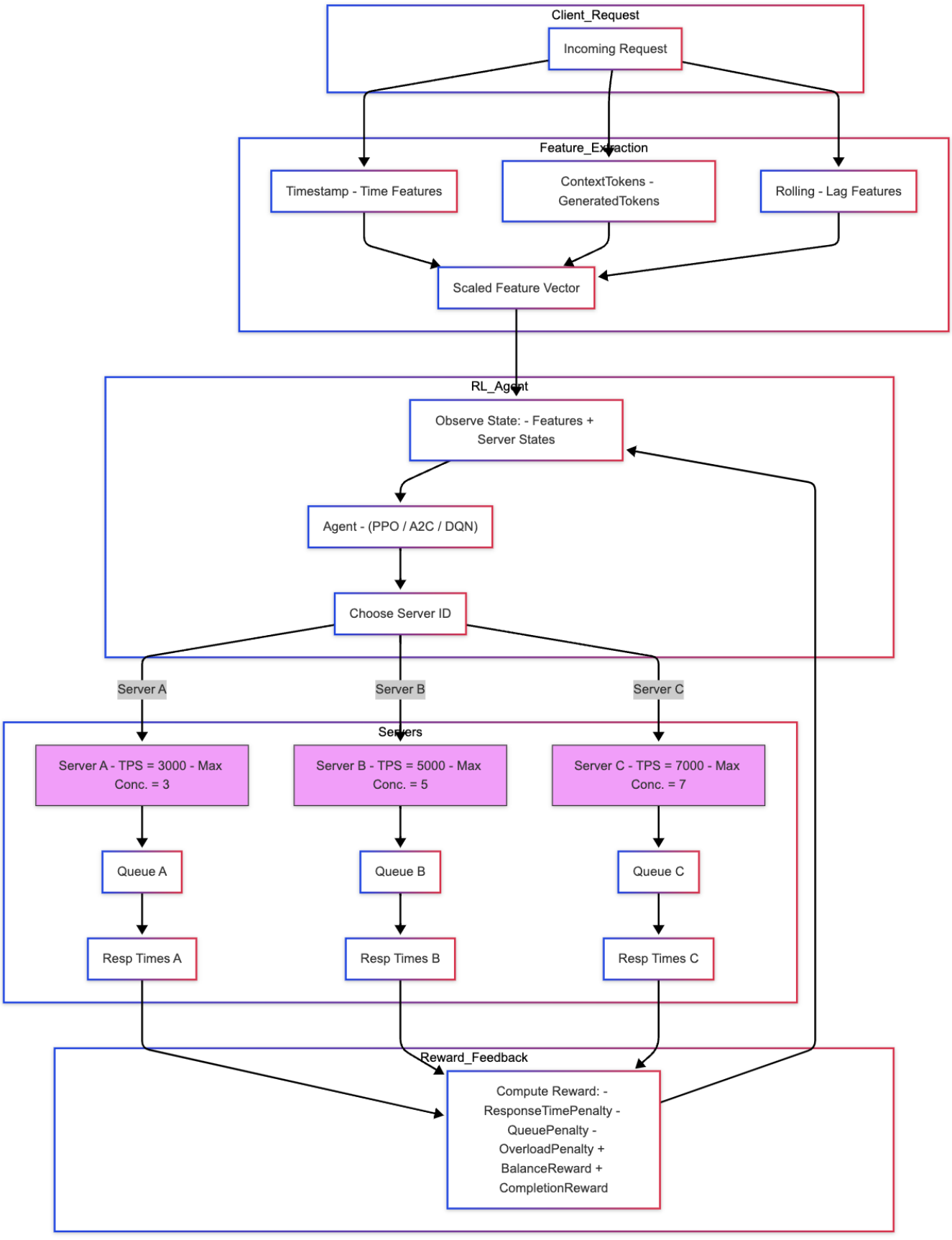

Fig. 10: System architecture diagram.